The Development Version Control and Visibility Subsystem

L. R. Hawley
DSN Data Systems Section

This article describes a prototype Development Version Control and Visibility Subsystem (DVCS). DVCS provides an implementation/management interface serving both the implementor and management. For management, DVCS monitors the production of design and source code. DVCS provides the implementor listings annotated with change bars, detects errors in the block structure of the design and indicates when standards requiring the use of structured programming constructs in the design of software are violated. It is operated by the Software Production and Management Control (SPMC) group of the Deep Space Network at the Jet Propulsion Laboratory in Pasadena, California. The DVCS will be used by the DSN to monitor the implementation of the Network Consolidation Project, a multiyear project at JPL.

I. Introduction

Modern software production methodologies promote the idea of an all-inclusive programming environment. Such an environment provides not only the tools required by programmers to implement a given software project, but includes mechanisms to monitor and manage the data base of code and documentation produced by the implementors. With regard to these management features of a programming environment, the DVCS addresses three main areas of current concerns: (1) monitoring the output of the implementation staff, (2) adherence to standard practices, and (3) the production of high-quality as-built software specification documentation.

Within the DSN organization is a Software Production and Management Control (SPMC) group. This group is chartered to:

(1) Monitor the production of software code and documentation in terms of quantity (lines of code).

(2) Produce final versions of software documentation describing implemented code.

(3) Provide management with reports on the status of software production.

(4) Other activities not directly connected with the DVCS such as archiving release versions, distribution, etc.

The Development Version Control and Visibility System (DVCS) is intended to automate, to some extent, items (1) through (3) above. The remainder of this paper is in four sections. Section II enumerates the goals and objectives of the DVCS. Sections III and IV describe its implementation and

current use. Section V provides conclusions and indicates future plans.

II. Goals and Objectives

A brief description of the software implementation process follows; goals and objectives of DVCS are then related to the functions that occur within the software generation process. First, a software engineer analyzes a Software Requirements Document (SRD) and produces a Software Design Document (SDD) describing the architecture of a software module to implement the specified functions. After review and approval, the implementation phase begins. The major deliverables of the implementation phase are:

(1) The source, object and other (e.g., job control) code necessary to generate a working system.

(2) A Software Specification Document (SSD) describing the as-built system. Within the SSD there is a detailed design section comprised of flowcharts or flowchart equivalents describing the software. This detailed design section may comprise roughly 90% of an SSD; this is clearly an area that can benefit from automation.

(3) Other documentation related to the use, test, and interface to other software.

As major goals, DVCS will benefit the following:

(1) Implementors: DVCS will foster the production of design prior to coding. A PDL language (flowchart equivalent) will generate design and derived documentation (e.g., variable cross reference) useful in the implementation process.

(2) SPMC: DVCS will extract PDL pseudocode from files containing both PDL and assembler/compiler statements and automate the production of design documentation. DVCS will measure the progress of an implementation by monitoring and reporting lines of code, lines of PDL, changed lines, etc. In addition, DVCS will verify that the design conforms to DSN standards for operational software.

(3) Management: DVCS provides a means of measuring the progress of a project. Automation of the measuring process ensures that management receives timely information (as supplied by SPMC, which does the measuring and reporting of a project's progress).

(4) End-Users: End use functions consist mainly of operations, sustaining and maintenance. These users should receive indirect benefits resulting from the consistent and orderly production of software code and documentation. The sustaining and maintenance functions derive the same direct benefits useful to the implementors when making changes to a software module. The database of code and documentation generated in the implementation phase is available to the software personnel involved with the sustaining and maintenance functions.

A major goal in the implementation of DVCS is to make a tool that is easily operable by SPMC personnel and will not require the use of a skilled programmer for its operation. The DVCS should be adaptable to different situations and flexible in its use. Lastly, it was desired to code the DVCS in the HAL/S programming language to gain experience using the DSN's standard real-time high level language.

III. Implementation

The implementation of the DVCS software utilizes design techniques to maximize the functionality of the component parts while reducing the interfaces between modules to a minimum. Essentially, this consists of separating the requirements for DVCS into modules that do one function completely, and nothing else. The data interfaces between modules are then checked for simplicity. The design is iterated until an acceptable level of module functionality and simplicity of module interfaces is achieved. The use of "bubble charts" allows the designer to describe a program as a set of functions (inside circles known as "bubbles") interconnected by lines representing data interfaces. Figure 1 shows a high level "bubble chart" for the DVCS. In concept the "bubbles" are separate tasks that may operate in parallel in a multi-tasking operating system. In practice, such a scheme is practical only if some form of pipelining is available to implement the data interface; that is, the output of one task is the input to the next task. The concept of "pipelining" can be visualized as a mechanism by which data flows through a pipe from one task to another. Unfortunately DVCS was not implemented on a computer that provided pipelining; it is, however, a useful conceptual tool. In place of a "pipeline" the current DVCS utilizes intermediate files.

In order to achieve maximum flexibility, DVCS is designed as a set of separate functions that may be combined in a desired order by the job control language that initiates execution of tasks on the computer. The major functions incorporated into the current prototype version of DVCS are:

(1) Initialize DVCS: This function creates an empty file for the summary data produced by DVCS78.

(2) DVCS78: This is the main function provided. It inputs the current and previous versions of a specified module's source code and outputs (1) a summary of the module including lines of code, lines of comments and

lines of PDL statements, (2) a listing of the input source annotated with "!" characters to indicate changes from one version to another, (3) PDL statements for a function that validates the design adheres to standards, and (4) writes extracted PDL statements onto a file consisting solely of PDL statements for processing by a PDL processor (a commercial product).

(3) Summarize Statistics: This function creates a totals summary of all module statistics that have been written to the summary file by DVCS78. Its output is a listing used by SPMC to report weekly progress.

(4) Validate Design: This function inputs PDL statements and checks that the design does not violate standard structured programming constructs. In addition, this function checks for errors in the block structure of the design. The output of this function is a Design Structure Analysis listing of violations of design structure standards. Typical design errors detected are the use of GOTO statements, loop exits from the middle of a loop, IFs without ENDIFs, and DOs without ENDs. Currently there is only one Validate Design module; in the future there may be other validation modules for other PDL languages.

IV. Current Usage

The prototype DVCS has recently been released for use by SPMC for a three-month evaluation period. Currently, SPMC personnel are being trained in the use of DVCS and specification of the job control statements required to initiate execution of DVCS. Eventually, the production version of DVCS will be utilized by the DSN in the implementation of the Network Consolidation Project (NCP), a multiyear project currently in the design stage at JPL.

It was previously mentioned that DVCS is coded in HAL/S, the DSN's standard high-level real-time language. There was some risk involved in selecting HAL/S since the DVCS is not a real-time application and the needs of real-time programs are significantly different from a "businesslike" program such as DVCS. On the positive side, the execution speed of DVCS is extremely fast. This was to be expected since the HAL/S compiler must produce highly optimized code for real-time programs. Other features of HAL/S that benefited the design process were (1) the standard structured programming constructs are implemented as primitives of the language, (2) HAL/S provides useful data structures such as strings, records, etc., (3) variable cross reference and other information derived from the source listing, (4) formatted "pretty printed" listings using indentation to indicate levels of nesting of conditional and iterative structured programming constructs. On the negative side, HAL/S does not provide for file I/O as primitives of the language (the programmer must implement code to open, read, write, position and close a file). Taken as a whole, HAL/S was significantly superior to the alternative languages normally available on the Modcomp computer (assembly and FORTRAN IV).

V. Conclusions

As indicated in Section I, the DVCS is one component of a total programming environment serving the needs of the implementors, management staff, and subsequent operations. DVCS is, in many ways, an implementation/management interface. It monitors the production of design and source code for reporting to management. DVCS also provides the implementor with listings annotated with change bars, detects errors in the block structure of the design and indicates when DSN standards require the use of structured programming constructs in the design of software. Future plans involve making the prototype into a production version and including additional functions to:

(1) Produce a magnetic tape to interface with a high-speed laser printer on another computer.

(2) Generate a change page listing indicating which pages of PDL listings changed (as a result of changes to the PDL).

(3) Additional functions determined desirable in the current evaluation phase.

References

1. Constantine, L, and Yourdon, E., *Structured Design*, Yourdon Press, New York, 1975.
2. Stenning, V., Froggatt, T., Gilbert, R., and Thomas, E., "The Ada Programming Environment: A Perspective," *Computer Magazine*, June 1981.
3. Kernighan, B., and Masinter, L, "The Unix Programming Environment," *Computer Magazine*, April 1981.

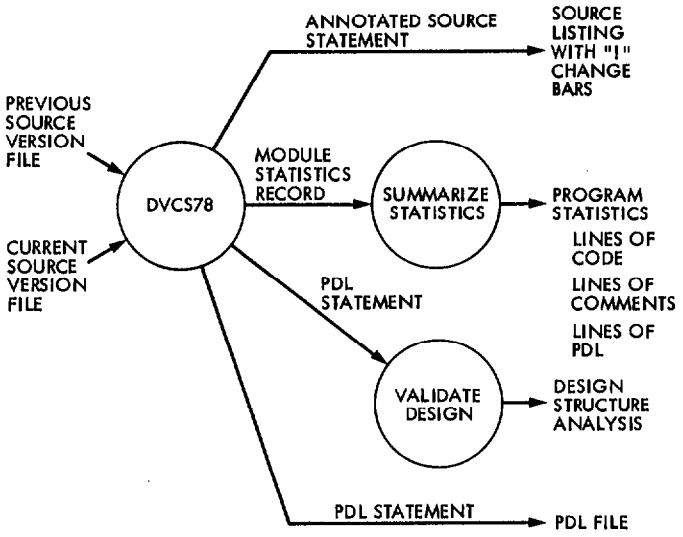

Fig. 1. DVCS bubble chart